1. 存儲的數據特點：*存储有序的、可重复的数据。*
2. 常用方法：（記住）

*增：add(Object obj)  
删：remove(int index) / remove(Object obj)  
改：set(int index, Object ele)  
查：get(int index)  
插：add(int index, Object ele)  
长度：size()  
遍历：① Iterator迭代器方式  
 ② 增强for循环  
 ③ 普通的循环*

1. 常用實現類：

*|----Collection接口：单列集合，用来存储一个一个的对象  
 |----List接口：存储有序的、可重复的数据。 -->“动态”数组,替换原有的数组  
 |----ArrayList：作为List接口的主要实现类；线程不安全的，效率高；底层使用*

*Object[] elementData存储:使用場景適合查找  
 |----LinkedList：对于频繁的插入、删除操作，使用此类效率比ArrayList高；底层*

*使用双向链表存储  
 |----Vector：作为List接口的古老实现类；线程安全的，效率低；底层使用Object[]*

*elementData存储(底層擴容為2倍)*

1. 源碼分析（難點）

*4.1 ArrayList的源码分析：使用場景適合查找  
 2.1 jdk 7情况下:底层创建了长度是10的Object[]数组,當某第次的add導致容量不夠,則默认情*

*况下，扩容为原来的容量的1.5倍，同时需要将原有数组中的数据复制到新的数组中。  
 ArrayList list = new ArrayList();//底层创建了长度是10的Object[]数组elementData  
 list.add(123);//elementData[0] = new Integer(123);  
 ...  
 list.add(11);//如果此次的添加导致底层elementData数组容量不够，则扩容。  
 默认情况下，扩容为原来的容量的1.5倍，同时需要将原有数组中的数据复制到新的数组中。  
 结论：建议开发中使用带参的构造器：ArrayList list = new ArrayList(int capacity):設*

*定初始長度  
 2.2 jdk 8中ArrayList的变化：初始并没有创建长度为10的数组,而是第一次调用add()时，底*

*层才创建了长度10的数组，并将元素添加至數組中,后续的添加和扩容操作与jdk 7 无异。  
 ArrayList list = new ArrayList();//底层Object[] elementData初始化为{}.并没有创*

*建长度为10的数组  
 list.add(123);//第一次调用add()时，底层才创建了长度10的数组，并将数据123添加到*

*elementData[0]  
 后续的添加和扩容操作与jdk 7 无异。  
 2.3 小结：jdk7中的ArrayList的对象的创建类似于单例的饿汉式，而jdk8中的ArrayList的对*

*象的创建类似于单例的懒汉式，延迟了数组的创建，节省内存。*

*4.2 LinkedList的源码分析：使用場景:適合頻繁增加刪除的部分,不適合查找  
 LinkedList list = new LinkedList(); 内部声明了Node类型的first和last属性，默认值为null  
 list.add(123);//将123封装到Node中，创建了Node对象。  
 其中，Node定义为：体现了LinkedList的双向链表的说法  
 private static class Node<E> {  
 E item;  
 Node<E> next; //存下一個的地址值  
 Node<E> prev; //存上一個的地址值  
 Node(Node<E> prev, E element, Node<E> next) {  
 this.item = element;  
 this.next = next;  
 this.prev = prev;  
 }  
 }*

*4.3 Vector的源码分析：*

*jdk7和jdk8中通过Vector()构造器创建对象时，底层都创建了长度为10的数组。  
在扩容方面，默认扩容为原来的数组长度的2倍。*

1. 存儲的元素要求：

添加的對象，所在的類要重寫equals()方法

*面试题：ArrayList、LinkedList、Vector三者的异同？  
 同：三个类都是实现了List接口，存储数据的特点相同：存储有序的、可重复的数据  
 不同：见上(第三部分 + 第四部分)*